Finding a Nash-equilibrium in noncooperative N-person games by solving a sequence of linear stationary point problems

Talman, A.J.J.; van den Elzen, A.H.

Published in:
ZOR: Zeitschrift für Operations Research: Methods and Models for Operations Research

Publication date:
1994

ZOR – Mathematical Methods of Operations Research (1994) 39:365–375

Finding a Nash Equilibrium in Noncooperative N-Person Games by Solving a Sequence of Linear Stationary Point Problems[1]

ANTOON VAN DEN ELZEN AND DOLF TALMAN

Department of Econometrics, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Abstract: In this paper we present an algorithm for finding a Nash equilibrium in a noncooperative normal form N-person game. More generally, the algorithm can be applied for solving a nonlinear stationary point problem on a simplotope, being the Cartesian product of several simplices. The algorithm solves the problem by solving a sequence of linear stationary point problems. Each problem in the sequence is solved in a finite number of iterations. Although the overall convergence cannot be proved, the method performs rather well. Computational results suggest that this algorithm performs at least as good as simplicial algorithms do.

For the special case of a bi-matrix game ($N = 2$), the algorithm has an appealing game-theoretic interpretation. In that case, the problem is linear and the algorithm always finds a solution. Furthermore, the equilibrium found in a bi-matrix game is perfect whenever the algorithm starts from a strategy vector at which all actions are played with positive probability.

Key Words: noncooperative game, Nash equilibrium, stationary point.

1 Introduction

In this paper we present an algorithm for computing a Nash equilibrium (NE) in a noncooperative N-person game in normal form. First we show that a Nash equilibrium of such a game is a stationary point of a multilinear function on a simplotope, being the Cartesian product of several unit simplices. The algorithm solves the stationary point problem of a continuous function on the simplotope and to find a Nash equilibrium it roughly works as follows. It starts from an arbitrary strategy vector in the simplotope. The multilinear function is linearized by taking the first-order Taylor expansion around that starting vector and extending it over the whole strategy space. Then the algorithm finds a stationary point on the simplotope for that linear function. If that strategy vector is a

[1] This research is part of the VF-program "Co-operation and Competition", which has been approved by the Netherlands Ministery of Education and Sciences.

0340–9422/94/39:3/365–375 $2.50

sufficient approximation of a Nash equilibrium for the original game, then the algorithm stops. Else the whole procedure is repeated but then from the strategy vector just found, and so on. This procedure is motivated by Mathiesen [1985a, b], [1987] who solves a Nonlinear Complementarity Problem (NLCP) by a sequence of Linear Complementarity Problems (LCP's). In practice this method works quite satisfactory, although global convergence is not guaranteed. Contrary to our method, his procedure may even fail to solve a specific LCP (see Mathiesen [1985b]).

Because our method solves a Sequence of Linear Stationary Point Problems we often refer to it as the SLSPP-algorithm. This SLSPP-algorithm is a generalization of the method presented in van den Elzen and Talman [1991] for finding an NE in a bi-matrix game. In that case the relevant function is linear and an exact solution is found.

The most well-known methods for finding an NE in an N-person game are the simplicial algorithms. For an exposition and some computational results we refer to Doup and Talman [1987]. At the end of this paper we compare the results obtained with the simplicial methods and the results we obtained with the procedure discussed here. We remark that in some sense the method sketched above works similar as a simplicial algorithm. Such a method finds in each round an exact solution for a piecewise linear approximation of the original function. Thus, then the function is linearized on each simplex of some simplicial subdivision, whereas here we take a linear approximation on the whole set. The advantage of simplicial algorithms is that their global convergence is guaranteed. However, especially in the later rounds simplicial algorithms may need more iterations to reach a solution. In practice it seems best to combine both methods by applying a simplicial algorithm during the first rounds, and later on the method discussed here. Concerning other methods we refer to the articles of Rosenmüller [1971] and Wilson [1971]. However, both articles are theoretical of nature and of hardly any use for practical implementation. Rosenmüller argues that each nondegenerated N-person game has an odd number of isolated Nash equilibria. The reasoning is similar to that in Lemke and Howson [1964] for bi-matrix games; there is a path connecting the constructed starting vector and an NE whereas the other Nash equilibria are pairwise connected by paths. However, he gives no procedure how to follow these paths. The latter is crucial because these paths are in general not linear. Wilson [1971] proves the existence of a path leading to an NE in an N-person game. This path finds in succession an equilibrium for each of certain related k-person games for k increasing from 1 to N. An equilibrium for the k-person game is then the starting point of a path leading to an NE of a $(k + 1)$-person game and so on. However, also here merely the existence of the path is proved and no method is given to follow it.

We stress the fact that our method can be used for other problems then the problem of finding a Nash equilibrium. In fact it can be used for finding a stationary point of an arbitrary continuous function on a simplotope.

The set-up of this paper is as follows. In Section 2 we introduce some notation and show that the set of Nash equilibria coincides with the solution set to a stationary point problem on a simplotope. Furthermore, we introduce the algorithm. Next, in Section 3 we present the formal steps of the algorithm. Finally, in Section 4 we give a game-theoretic interpretation of the algorithm and present an example. Furthermore, we compare the performance of the SLSPP-algorithm with that of the simplicial algorithms.

2 Nash Equilibria as Solutions to a Stationary Point Problem

We consider noncooperative games with a finite number of players each having a finite set of actions. The payoffs to a player are listed in a tensor. Each element in the tensor indicates the payoff to that player when a specific set of actions is played by the players.

Let us introduce some notation. The number of players equals N and the players are indexed by $j, j \in \{1, \ldots, N\}$. The set $\{1, \ldots, N\}$ is often denoted as I_N. Player j has n_j actions. Action k of player j is indicated as (j, k). The set of actions of player j, i.e. the set $\{(j, 1), \ldots, (j, n_j)\}$ is denoted by $I(j)$. The set of all actions in the game is often denoted by I, i.e. $I = \bigcup_{j \in I_N} I(j)$. The payoffs to player j are listed in a tensor $A^j, j \in I_N$. The payoff to player j in case each player $i \in I_N$ plays action (i, ℓ_i) is denoted by $A^j(\ell_1, \ldots, \ell_{j-1}, \ell_j, \ell_{j+1}, \ldots, \ell_N)$.

A strategy of player $j, j \in I_N$, is represented by a vector $x_j = (x_{j1}, \ldots, x_{jn_j})^\top$ in $S^{n_j-1} := \{x_j \in \mathbb{R}_+^{n_j} | \sum_{k=1}^{n_j} x_{jk} = 1\}$. The number x_{jk}, $k \in \{1, \ldots, n_j\}$, is then the probability with which player j plays his k-th action at strategy x_j. Since the components of x_j are all nonnegative and sum up to one, x_j is indeed a vector of probabilities. We call S^{n_j-1} the strategy space of player j. In case actions are played with probability one we speak about pure strategies. The strategy space of the game is then the simplotope S obtained by taking the Cartesian product of the strategy spaces of the players, i.e. $S = S^{n_1-1} \times \cdots \times S^{n_N-1}$. An element $x = (x_1, \ldots, x_N)$ in S denotes a strategy vector of the game with x_j the strategy played by player $j \in I_N$.

The marginal expected payoff function z is a function from S to $\prod_{j=1}^N \mathbb{R}^{n_j}$ defined by $z(x) = (z_1(x), \ldots, z_N(x))$, with

$$z_{jk}(x) = \sum_{\ell_1=1}^{n_1} \cdots \sum_{\ell_{j-1}=1}^{n_{j-1}} \sum_{\ell_{j+1}=1}^{n_{j+1}} \cdots \sum_{\ell_N=1}^{n_N} A^j(\ell_1, \ldots, \ell_{j-1}, k, \ell_{j+1}, \ldots, \ell_N) \prod_{i \neq j} x_{i\ell_i},$$

$$(j, k) \in I(j), j \in I_N \ .$$

Thus $z_{jk}(x)$ is the payoff to player j in case he plays his k-th pure stretegy while each other player i plays x_i. Now a Nash equilibrium is defined as a strategy vector at which no player can improve upon his situation by unilaterally deviating from his strategy. Thus, $x^* = (x_1^*, \ldots, x_N^*) \in S$ is a Nash equilibrium if

$$x_j^\top z_j(x^*) \leq (x_j^*)^\top z_j(x^*),\ x_j \in S^{n_j - 1},\ j \in I_N\ . \tag{2.1}$$

The number $(x_j^*)^\top z_j(x^*)$ equals the expected payoff of player $j \in I_N$ at the strategy vector x^*. Thus, at a Nash equilibrium x^* it is optimal for each player j to play x_j^*. System (2.1) shows that x^* solves the stationary point problem of the function z on S. Indeed the set of Nash equilibria corresponds to the solution set of this stationary point problem.

Because of the linearity of $x_j^\top z_j(x^*)$ in x_j we only have to check (2.1) for the vertices of S^{n_j-1}, i.e. for the pure strategies. Thus, x^* is a Nash equilibrium if and only if $z_{jk}(x^*) \leq (x_j^*)^\top z_j(x^*)$ for all $(j, k) \in I$. From this it is straightforward to derive that x^* is a Nash equilibrium if and only if

$$z_{jk}(x^*) = \max_h z_{jh}(x^*) \text{ when } x_{jk}^* > 0,\ (j, k) \in I\ . \tag{2.2}$$

In general we say that an action k of player j is optimal at strategy x when $z_{jk}(x) = \max_h z_{jh}(x)$ and that at x player j is in equilibrium when at x all nonoptimal actions of j are played with probability zero. The interpretation of (2.2) is that at a Nash equilibrium the actions being played with positive probability are all optimal and hence that all players are in equilibrium.

How does the algorithm work to find a strategy vector x^* obeying (2.2)? First we choose an arbitrary starting vector $v = (v_1, \ldots, v_N)$ in S. Then we linearize z around v to obtain the function $z^v\colon S \to \mathbb{R}^n$, defined by

$$z^v(x) = z(v) + Dz(v)(x - v)\ , \tag{2.3}$$

where $n = \sum_{j=1}^N n_j$, and $Dz(v)$ is the $(n \times n)$-matrix of derivatives of z at v. More precisely, the (p, q)-th element of $Dz(v)$ equals $\delta z_{jk}(v)/\delta x_{ih}$, where $p = \sum_{\ell=1}^{j-1} n_\ell + k$ and $q = \sum_{\ell=1}^{i-1} n_\ell + h$. Observe that this element is zero if $(i, h) \in I(j)$. Note also that $z(v)$, x, and v are represented as vectors in $\mathbb{R}^n$. Now we apply a generalized version of the algorithm presented in van den Elzen and Talman [1991], to find an NE for the game with marginal expected payoff function z^v. If that strategy vector is a close enough approximation of an NE for the original game then the algorithm stops. Else the procedure is restarted from the strategy vector just found.

So, the algorithm searches for an equilibrium for the game with marginal payoff function z^v. It does so by generating from v a path of strategy vectors $x = (x_1, \ldots, x_N)$ in S obeying for $(j, k) \in I$,

$$\begin{aligned} x_{jk} &= b(x, v)v_{jk} \text{ if } z_{jk}^v(x) < \max_h z_{jh}^v(x) \\ x_{jk} &\geq b(x, v)v_{jk} \text{ if } z_{jk}^v(x) = \max_h z_{jh}^v(x)\ , \end{aligned} \tag{2.4}$$

where $0 \leq b(x, v) := min_{(j,h)}\{x_{jh}/v_{jh} | v_{jh} > 0\} \leq 1$.

Observe that the strategy vector v obeys (2.4) with $b(x, v) = 1$. Also any Nash equilibrium x^* for the game with marginal payoff function z^v obeys (2.4) with $b(x^*, v) = 0$ or with $v_{jk} = 0$ for all indices (j, k) for which $z_{jk}^v(x^*) < \max_h z_{jh}^v(x^*)$. In both cases $x_{jk}^* = b(x^*, v)v_{jk}$ is equal to zero when at x^* action (j, k) is nonoptimal. Under some nondegeneracy condition the set of points satisfying (2.4) contains a piecewise linear path, P, from v to a Nash equilibrium related to z^v. That Nash equilibrium is an approximation for a Nash equilibrium for the original game. The path P will be followed by the algorithm as described in the next section.

The notion of nondegeneracy will be made precise further on, but it is for example required that at $x = v$ it must hold that $\max_h z_{jh}^v(x)$, $j \in I_N$, is attained for a unique index. Thus, at the starting strategy vector v each player has only one optimal action. Suppose these maxima are attained for the actions (j, k_j), $j \in I_N$. Clearly, from v, along the path P, $b(x, v)$ must decrease from 1. Thus, according to (2.4) initially vectors x in S are generated such that all the x_{jk} with $k \neq k_j, j \in I_N$, are relatively decreased ($x_{i\ell} = b(x, v)v_{i\ell}$ for $(i, \ell) \notin \{(j, k_j), j \in I_N\}$), while every $x_{jk_j}, j \in I_N$, is increased to keep x_j in $S^{n_j - 1}$. This is continued till either $b(x, v)$ becomes 0 and a Nash equilibrium for z^v is reached, or a strategy vector x is reached at which $z_{jk}^v(x) = z_{jk_j}^v(x)$ for some (j, k), $k \neq k_j$. Then x_{jk} is also relatively increased. In general, the algorithm generates strategy vectors x such that all the x_{jk}/v_{jk}, related to the indices (j, k) for which $z_{jk}^v(x) < \max_h z_{jh}^v(x)$, are minimal, i.e. equal to $b(x, v)$ ($x_{jk} = 0$ if $v_{jk} = 0$). As soon as one of these components of $z^v(x)$, say $z_{j\ell}^v(x)$, becomes equal to $\max_h z_{jh}^v(x)$, then $x_{j\ell}/v_{j\ell}$ is increased from $b(x, v)$ ($x_{j\ell}$ is increased from zero if $v_{j\ell} = 0$), while keeping $z_{j\ell}^v(x)$ maximal for j. On the other hand, if a vector x is generated such that x_{jr}/v_{jr} for some (j, r) with $z_{jr}^v(x) = \max_h z_{jh}^v(x)$ becomes minimal, i.e. equal to $b(x, v)$ (x_{jr} becomes 0 if $v_{jr} = 0$), then vectors y are generated with y_{jr} equal to $b(y, v)v_{jr}$ while $z_{jr}^v(y)$ is decreased from $\max_h z_{jh}^v(y)$.

3 The Procedure

The algorithm is a complementary pivoting procedure. To implement it we have to write (2.4) as a system of linear equations. Observe that for each vector x satisfying (2.4) there is at least one set $T \subset I$ with $T_j = T \cap I(j) \neq \emptyset$ for all j such that $x_{jk} \geq bv_{jk}$ and $z_{jk}^v(x) = \beta_j$ for all $(j, k) \in T$ while $z_{ih}^v(x) \leq \beta_i$ and $x_{ih} = bv_{ih}$ for all $(i, h) \notin T$, where $b = b(x, v)$ and $\beta_j = \max_h z_{jh}^v(x)$. From this observation we obtain that the procedure generates for a sequence of subsets T of I, starting from $x = v$, strategy vectors x in S, such that

$$x = bv + \sum_{(j,k) \in T} \lambda_{jk} e(j, k)$$

and (3.1)

$$z^v(x) = -\sum_{(i,h)\notin T} \mu_{ih} e(i,h) + \sum_{j=1}^{N} \beta_j \bar{e}_j \ ,$$

with $\sum_{(j,k)\in T_j} \lambda_{jk} = 1 - b$ for $j \in I_N$, $\lambda_{jk} \geq 0$ for $(j,k) \in T$, $\mu_{ih} \geq 0$ for $(i,h) \notin T$, $b \in [0, 1]$. The vector $e(j, k)$ denotes the $(\sum_{i=1}^{j-1} n_i + k)$-th unit vector in $\mathbb{R}^n$ and $\bar{e}_j$ denotes the vector in $\mathbb{R}^n$ with ones on the places $\sum_{\ell=1}^{j-1} n_\ell + k$, $k \in \{1, \ldots, n_j\}$, and zeros elsewhere.

So, we obtain the linear system (3.1) from (2.4) by introducing slacks for each inequality. Next, we substitute the first system of equations in (3.1) into the second system and together with (2.3) we obtain the system of linear equations

$$bDz(v)v + \sum_{(j,k)\in T} \lambda_{jk} D^{jk} z(v) + \sum_{(i,h)\notin T} \mu_{ih} e(i,h) - \sum_{j\in I_N} \beta_j \bar{e}_j = Dz(v)v - z(v) \quad (3.2)$$

with $b + \sum_{(j,k)\in T_j} \lambda_{jk} = 1$, for $j \in I_N$.

Here $D^{jk}z(v)$ denotes the $(\sum_{i=1}^{j-1} n_i + k)$-th column of the matrix $Dz(v)$.

We obtain the final system by substituting $b' = 1 - b$ and for $j \in I_N$, λ_{jk_j} by $b' - \sum_{k\neq k_j} \lambda_{jk}$ where k_j is the index for which $z_{jk_j}(v) = \max_\ell z_{j\ell}(v)$. In this way, we obtain a system of n equations with $n + 1$ variables. More precise, we get the system of equations

$$b'\left(\sum_{j\in I_N} D^{jk_j} z(v) - Dz(v)v\right) + \sum_{\substack{(j,k)\in T\\ k\neq k_j}} \lambda_{jk}(D^{jk}z(v) - D^{jk_j}z(v))$$

$$+ \sum_{(i,h)\notin T} \mu_{ih} e(i,h) - \sum_{j\in I_N} \beta_j \bar{e}_j$$

$$= -z(v) \ , \quad (3.3)$$

with additional restrictions $\lambda_{jk} \geq 0$ for $(j,k) \in T_j \backslash \{(j, k_j)\}$, $\sum_{k\neq k_j} \lambda_{jk} \leq b'$ for $j \in I_N$, $0 \leq b' \leq 1$, and $\mu_{ih} \geq 0$ for $(i,h) \notin T$. Observe that (3.3) is equivalent to system (3.7) in van den Elzen and Talman [1991] in case $N = 2$ because in that case

$$Dz(v) = \begin{bmatrix} 0 & A \\ B^T & 0 \end{bmatrix} .$$

A solution to (3.3) is denoted as $(b', \lambda, \mu, \beta)$ with $\lambda = (\lambda_{jk})_{(j,k)\in T}$, $\mu = (\mu_{ih})_{(i,h)\notin T}$, and $\beta = (\beta_1, \ldots, \beta_N)$. The algorithm is a complementary pivoting procedure operating in system (3.3), with the variables in λ and μ being complementary. So, if λ_{jk} for some $(j, k) \in T$ becomes zero then μ_{jk} is increased from zero and vice versa. To guarantee convergence we need that not more than one variable becomes zero at the same time.

Assumption 3.1: (Nondegeneracy Assumption). At each solution $(b', \lambda, \mu, \beta)$ of (3.3) at most one of the constraints $0 \leq b' \leq 1$, $\lambda_{jk} \geq 0$ for $(j, k) \in T^1 = T \backslash \{(j, k_j), j \in I_N\}$, $b' \geq \sum_{(j,k) \in T_j^1} \lambda_{jk}$, $\mu_{ih} \geq 0$ for $(i, h) \notin T$, is binding, unless $b' = 1$ or $v_{ih} = 0$ for all $(i, h) \notin T$.

The assumption above implies that at v each player should have only one optimal action. If for example $z_{j\ell}(v) = z_{jk_j}(v) = \beta_j$ for some index ℓ then two constraints would hold at the start ($b' = 0$ and $\mu_{j\ell} = 0$).

The formal steps of the procedure are the following.

Step 0 [Initialization]

Choose an arbitrary vector v in S. If v is a Nash equilibrium then the algorithm stops. Else choose a measure of inaccuracy δ, a maximum number of rounds $\bar{t}$, and set t equal to 1.

Step 1

If $t > \bar{t}$ then the algorithm stops and no approximate Nash equilibrium is found within $\bar{t}$ rounds. Else construct the first-order Taylor expansion z^v of z at v as in (2.3). Calculate for $j \in I_N$ the unique index (j, k_j) for which $z_{jk_j}(v) = \max_h z_{jh}(v)$. Furthermore, set $T^1 = \emptyset$, $b' = 0$, $\beta_j = z_{jk_j}(v)$ for $j \in I_N$, $\mu_{ih} = \beta_i - z_{ih}(v)$ for $h \neq k_i$ and $i \in I_N$. Increase b' from 0 in (3.3) and go to Step 2.

Step 2

a. If b' becomes 1 then let the solution of (3.3) be $(1, \lambda^*, \mu^*, \beta^*)$. The vector x^*, with $x^* = \sum_{(j,k) \in T} \lambda^*_{jk} e(j, k)$ is a solution to the SPP of z^v on S (cf. (2.2)). If $\max_{(j,k)}(z_{jk}(x^*) - (x_j^*)^\top z_j(x^*)) < \delta$ then x^* is an approximate Nash equilibrium and the algorithm stops after round t. Else t becomes $t + 1$ and return to Step 1 with v equal to x^*.
b. If λ_{jk} becomes 0 for some $(j, k) \in T^1$ then T^1 becomes $T^1 \backslash \{(j, k)\}$ and go to Step 3a.
c. If $\sum_{(j,k) \in T_j^1} \lambda_{jk}$ becomes equal to b' for some $j \in I_N$ then λ_{jk_j} becomes 0. Go to Step 3b.
d. If μ_{ih} becomes zero for some $(i, h) \notin T$ then go to Step 4.

Step 3

a. Increase the complementary variable μ_{jk} from zero by pivoting into system (3.3) the column $e(j, k)$. Return to Step 2.

b. Substitute the largest λ_{jk}, say $\lambda_{j\ell}$, by $b' - \sum_{(j,h) \in T_j, h \neq k_j, \ell} \lambda_{jh}$. Increase μ_{jk_j} from zero by pivoting the related column into system (3.3). T^1 becomes $T^1 \backslash \{(j, \ell)\}$, k_j becomes ℓ, and return to Step 2.

Step 4

a. If $v_{jk} = 0$ for all $(j, k) \notin T \cup \{(i, h)\}$ then let the solution be $(b'^*, \lambda^*, \mu^*, \beta^*)$. The vector $x^* = (1 - b'^*)v + \sum_{(j,k) \in T} \lambda^*_{jk} e(j, k)$ is a solution to the SPP of z^v on S (cf. (2.2)). If $\max_{(j,k)}(z_{jk}(x^*) - (x^*_j)^{\mathrm{T}} z_j(x^*)) < \delta$ then x^* is an approximate Nash equilibrium and the algorithm stops after round t. Else t becomes $t + 1$ and return to Step 1 with v equal to x^*.
b. Otherwise increase the complementary variable λ_{ih} from zero by pivoting its related column into system (3.3). T^1 becomes $T^1 \cup \{(i, h)\}$ and return to Step 2.

4 Game Theoretic Interpretation and Computation

The procedure presented in Section 3 has an intuitive game-theoretic interpretation for the game with marginal expected payoff function z^v. In case $N = 2$ the function z^v corresponds with z and the interpretation is even more appealing.

The procedure starts by increasing the probabilities with which at v the optimal actions are played whereas the probabilities related to the other actions are decreased relatively equally (b decreases from 1 and λ_{jk_j} increases from 0 for $j \in I_N$). With relatively we mean relative to the initial probabilities. In general the algorithm generates strategy vectors at which the relative probabilities related to the nonoptimal actions are all equal and minimal whereas they are zero if the probability is initially zero. As soon as a strategy vector is generated at which an action that was nonoptimal becomes optimal ($\mu_{i\ell}$ becomes zero for some $(i, \ell) \notin T$) then that action is kept optimal whereas the related relative probability is allowed to increase from the minimum ($\lambda_{i\ell}$ is increased). On the other hand, if a vector is generated at which the probability related to an optimal action becomes relatively minimal (λ_{jk} becomes zero for some $(j, k) \in T$) or zero if v_{jk} is zero, then the probability is kept relatively minimal (zero if v_{jk} is zero) and the action is made nonoptimal (μ_{jk} is increased from zero).

We illustrate the working of the algorithm by an example.

Example 4.1: We consider a noncooperative 3-person game, in which each player has two actions. The payoffs are listed below.

	(1, 1)			(1, 2)	
	(2, 1)	(2, 2)		(2, 1)	(2, 2)
(3, 1)	(1, 3, −2)	(2, 1, 4)	(3, 1)	(1, −3, 2)	(1, 1, −3)
(3, 2)	(−1, −2, 3)	(4, 5, −6)	(3, 2)	(1, −3, 4)	(−5, 1, −2)

The left matrix entails the payoffs in case player 1 plays his first action, the right matrix corresponds to strategy vectors at which player 1 plays his second action. Each entry in the matrix consists of three elements corresponding to the payoffs for each player. For example, if player 1 plays (1, 2), player 2 plays (2, 2) and 3 plays (3, 1) then player 1 and 2 gets a payoff of 1 whereas player 3 gets -3.

Let us apply our algorithm starting from $v = ((1/2, 1/2), (1/2, 1/2), (1/2, 1/2))$. Linearizing the marginal payoff function z at v gives

$$z^v(x) = \begin{bmatrix} 1.5 \\ -0.5 \\ -1.25 \\ 2 \\ 0.25 \\ -0.25 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 3 & 1.5 & 1.5 \\ 0 & 0 & 1 & -2 & 1 & -2 \\ 0.5 & -3 & 0 & 0 & 0 & -2.5 \\ 3 & 1 & 0 & 0 & 1 & 3 \\ 1 & -0.5 & 0 & 0.5 & 0 & 0 \\ -1.5 & 1 & 3.5 & -4 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{11} - 0.5 \\ x_{12} - 0.5 \\ x_{21} - 0.5 \\ x_{22} - 0.5 \\ x_{31} - 0.5 \\ x_{32} - 0.5 \end{bmatrix}.$$

At $x = v$, $z^v(x)$ equals $z(v)$ and the optimal action for player 1 is (1, 1), for player 2 action (2, 2), for player 3 action (3, 1). Thus, from the start the related probabilities are increased by increasing b' from zero (Step 1). After one step a stationary point for z^v is found because b' becomes 1 (Step 2a). It is easily checked that the vector $\bar{x} = ((1, 0), (0, 1), (1, 0))$ is indeed a Nash equilibrium related to the marginal payoff function z^v. However, $z(\bar{x})$ equals $(2, 1, 3, 1, 4, -6)^\top$, i.e. $\bar{x}$ is not a Nash equilibrium for the original game because player 2 is not in equilibrium at $\bar{x}$. In the second round we linearize z at $v = \bar{x}$ and obtain

$$z^v(x) = \begin{bmatrix} 2 \\ 1 \\ 3 \\ 1 \\ 4 \\ -6 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 & 2 & 2 & 4 \\ 0 & 0 & 1 & 1 & 1 & -5 \\ 3 & -3 & 0 & 0 & 3 & -2 \\ 1 & 1 & 0 & 0 & 1 & 5 \\ 4 & -3 & -2 & 4 & 0 & 0 \\ -6 & -2 & 3 & -6 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{11} - 1 \\ x_{12} \\ x_{21} \\ x_{22} - 1 \\ x_{31} - 1 \\ x_{32} \end{bmatrix}.$$

Table 4.1. Computational results obtained for three noncooperative more-person games. LP denotes the number of linear programming pivoting steps whereas v denotes the number of rounds

Game	Simplicial algorithms		SLSPP algorithm	
	LP	v	LP	v
1	35	4	28	4
2	14	1	6	2
3	14	3	14	3

At the start we have that $\beta_1 = z_{11}(v) = 2$, $\beta_2 = z_{21}(v) = 3$, and $\beta_3 = z_{31}(v) = 4$. Furthermore $\mu_{12} = 1$, $\mu_{22} = 2$, and $\mu_{32} = 10$. Then b' is increased from 0 (Step 1). If b' becomes 2/3 then μ_{32} becomes 0 (Step 2d). For the solution to (3.3) it further holds that $\beta = (\beta_1, \beta_2, \beta_3) = (4/3, 3, 0)$, $\mu_{12} = 1/3$, $\mu_{22} = 2$. Thus, now λ_{32} is increased (Step 4) till 2/9 when μ_{22} becomes zero (Step 2d). For the solution to (3.3) we have $\beta = (16/9, 17/9, 0)$, $\mu_{12} = 19/9$, $\lambda_{31} = 4/9$, $b' = 2/3$ with corresponding vector $x^* = ((1, 0), (2/3, 1/3), (7/9, 2/9))$. Since $v_{jk} = 0$ for all (j, k) such that $\mu_{jk} > 0$, x^* is an equilibrium for z^v (Step 4a). It is easily verified that x^* is also a Nash equilibrium for the original game.

Finally we compare the speed of our algorithm with that of simplicial algorithms. We applied our algorithm to the three games given in Doup and Talman [1987] and compared the results with those given in Doup [1988, ch. 11]. Three different simplicial algorithms are applied to the games. In Table 4.1 we compare our results with the best results obtained by the simplicial algorithms. It turns out that our algorithm is at least as good.

References

Doup TM (1988) Simplicial algorithms on the simplotope. Lecture Notes in Economics and Mathematical Systems 318, Springer-Verlag Berlin

Doup TM, Talman AJJ (1987) A new variable dimension simplicial algorithm to find equilibria on the product space of unit simplices. Mathematical Programming 37:319–355

Elzen AH van den, Talman AJJ (1991) A procedure for finding nash equilibria in bi-matrix games. ZOR-Methods and Models of Operations Research 35:27–43

Lemke CE, Howson JT (1964) Equilibrium points of bi-matrix games. SIAM Journal of Applied Mathematics 12:413–423

Mathiesen L (1985a) Computational experience in solving equilibrium models by a sequence of linear complementarity problems. Operations Research 33:1225–1250

Mathiesen L (1985b) Computation of economic equilibria by a sequence of linear complementarity problems. Mathematical Programming Study 23:144–162

Mathiesen L (1987) An algorithm based on a sequence of linear complementarity problems applied to a Walrasian equilibrium model: An example. Mathematical Programming 37:1–18

Rosenmüller J (1971) On a generalization of the Lemke-Howson algorithm to noncooperative N-person games. SIAM Journal of Applied Mathematics 21:73–79

Wilson R (1971) Computing equilibria of N-person games. SIAM Journal of Applied Mathematics 21:80–87

Received: October 1992
Revised version received: July 1993